UNIVERSITÉ DE FRANCHE-COMTÉ

IUT BESANÇON VESOUL

SD

SAÉ 1.02

Écriture et lecture de fichiers de données

Encadrée par :

Dorine Tabary

Réalisé par :

Ferrand Soki

Yasmine Taraza

Salma Alahyan

Table des Matières

1. Introduction

L’objectif de cette SAÉ est de concevoir des programmes permettant de traiter des fichiers contenant des informations d’étudiants. Ce traitement inclut l’extraction et la transformation des données pour les rendre utilisables par les professeurs.

Les données fournies comprennent 303 fiches d’étudiants, chacune correspondant à un fichier distinct. Chaque fiche contient des informations personnelles et des erreurs rencontrées par les étudiants lors de leurs travaux en programmation.

Ce projet permet de développer des compétences en manipulation de fichiers, en extraction de données et en développement de Python respectant des spécifications précises.

Ce rapport se propose de documenter en détail les travaux réalisés dans le cadre des lots 1 et 2 du projet, en mettant en lumière les formats de données utilisés, les choix techniques et méthodologiques effectués, ainsi que les résultats obtenus.

Le projet avait pour objectif principal de répondre à deux problématiques spécifiques : d'une part, la gestion des informations des étudiants, notamment leurs adresses électroniques, et d'autre part, l'analyse et la visualisation des erreurs commises en programmation. Ces solutions visent à faciliter le travail des enseignants et à offrir des outils pratiques pour les étudiants. Ce rapport explique les moyens mis en œuvre pour atteindre ces objectifs, discute les résultats obtenus, et justifie les choix technologiques et méthodologiques adoptés.

2. Justification des choix techniques

Formats utilisés

- **Editeur de texte (Word) :** Ce logiciel permet de créer des documents bien structurés, faciles à réviser et prêts à être partagés avec des collaborateurs. De plus, Word garantit une compatibilité universelle, ce qui assure que le rapport peut être ouvert et visualisé correctement par tous les destinataires.
- **Fichier md :** Il permet de créer des documents clairs et bien structurés avec une syntaxe simple. Markdown rend également les fichiers facilement lisibles et modifiables dans divers environnements, ce qui est crucial pour la documentation partagée.
- **CSV :** Utilisé pour stocker des données structurées, telles que des listes et des tableaux, sous forme de texte. Ils sont compatibles avec de nombreux logiciels, comme Excel, ce qui permet de les importer et de les exporter facilement. Leur format simple rend les données faciles à lire et à modifier.
- **Excel (XLSX) :** Facilite la création de tableaux, et de graphiques, rendant les données plus lisibles, compréhensibles et faciles à gérer.

Outils de Communication

- **GitHub :** Utilisé pour la communication et la collaboration car il offre des fonctionnalités robustes de gestion de version, facilitant le suivi des modifications au code. Il fournit aussi des outils de documentation comme les fichiers README.

Langage

- **Python :** Choisi pour sa simplicité, sa flexibilité et ses bibliothèques puissantes (openpyxl, collections) et ses modules (os, csv, workbook, counter).

3. Description des programmes

Lot 1 :

- **MailingList**

- **Objectif :** Extraire des informations spécifiques (NOM, PRÉNOM, et EMAIL) à partir de fichiers texte stockés dans un répertoire nommé data. Les données extraites sont ensuite organisées et sauvegardées dans un fichier CSV nommé mailingList.csv
- **Détails :**

Importation des modules

- **Os :** Permet de parcourir et de manipuler le système de fichiers, comme accéder aux fichiers d'un répertoire spécifique.
- **Csv :** Gère la création et la manipulation des fichiers au format CSV, utilisé ici pour structurer les données extraites.

Définition de la fonction extraire_valeur_apres_mot_cle

- **But :** Extraire les valeurs associées à des champs spécifiques (NOM, PRÉNOM, EMAIL) dans chaque ligne d'un fichier texte.
- Localiser la position où le mot-clé (comme NOM) apparaît.
- Ignorer les espaces ou tabulations suivant le mot-clé.
- Extraire le contenu jusqu'à la fin de la ligne ou jusqu'à rencontrer un retour à la ligne.

Définition des chemins et du fichier de sortie

- **Directory :** Chemin du répertoire contenant les fichiers source (ici, nommé data).
- **Output_csv :** Nom du fichier CSV qui contiendra les résultats (par défaut : mailingList.csv).
- **entete :** Liste des colonnes pour le fichier CSV : ['NOM', 'PRENOM', 'MAIL_ETUDIANTS'].

Création et écriture dans le fichier CSV

- **Ouverture :** Le fichier CSV est ouvert en mode écriture ('w') avec encodage UTF-8 et sans saut de ligne additionnel (newline='').
- **Écriture des en-têtes :** Les colonnes définies dans entete sont ajoutées au début du fichier CSV.

Parcours des fichiers

- La fonction **os.listdir(directory)** liste tous les fichiers dans le dossier data.
- Pour chaque fichier, on vérifie qu'il s'agit bien d'un fichier texte avec **os.path.isfile(f)**.

Lecture des fichiers

- Les fichiers texte sont ouverts en mode lecture ('r'), et leurs lignes sont parcourues.
- Trois variables (nom, prenom, email) servent à stocker les informations extraites.

Extraction des informations nécessaires

Pour chaque ligne d'un fichier texte :

- Si elle commence par NOM, PRÉNOM ou EMAIL, la fonction extraire_valeur_apres_mot_cle extrait la valeur correspondante.

Sauvegarde des données dans le fichier CSV

- Si les trois champs (nom, prenom, email) sont trouvés pour un fichier, une nouvelle ligne est ajoutée dans le CSV avec ces données.

Fin du programme

- Un message est affiché une fois l'extraction terminée, confirmant que les données ont été écrites dans le fichier mailingList.csv.

Infos_Etudiants

- **Objectif :** Ce programme extrait des informations clés (NOM, PRÉNOM, EMAIL) et recense les erreurs courantes (SyntaxError, TypeError, etc.) à partir de fichiers texte stockés dans un répertoire data. Les résultats sont organisés dans un fichier Excel nommé Infos_Etudiants1.xlsx.

- **Détails :**

Importation des modules

❖ **openpyxl** : Gère la création et la manipulation des fichiers Excel (XLSX).
❖ **collections.Counter** : Facilite le comptage des occurrences d'erreurs spécifiques.

Initialisation des structures de données

❖ **Répertoire source :** data, contenant les fichiers texte.
❖ **Fichier de sortie :** Infos_Etudiants1.xlsx.
❖ **Classeur Excel :** Créé avec deux feuilles :

- Etudiants : Contient les informations NOM, PRÉNOM, et EMAIL.
- Erreurs : Liste les erreurs rencontrées et leur fréquence.

❖ **Dictionnaire erreurs :** Utilisé pour compter les occurrences des différentes erreurs.

Parcours des fichiers

❖ Le programme parcourt tous les fichiers dans data avec **os.listdir**.
❖ Pour chaque fichier :
  - Vérifie qu'il s'agit d'un fichier valide avec **os.path.isfile**.
  - Ouvre le fichier en mode lecture et lit toutes les lignes.

Extraction des données des étudiants

❖ Parcourt chaque ligne du fichier pour identifier et extraire :

  - **NOM :** Utilise la fonction extraire_valeur_apres_mot_cle si la ligne commence par NOM.
  - **PRÉNOM :** Idem pour PRENOM.
  - **EMAIL :** Idem pour EMAIL.

❖ Si les trois champs sont présents, ils sont ajoutés comme nouvelle ligne dans la feuille Etudiants.

Détection et comptage des erreur

- Recherches des types d'erreurs suivants dans chaque ligne du fichier:
  - SyntaxError
  - TypeError
  - NameError
  - IndentationError
- Chaque occurrence d'erreur est comptabilisée à l'aide du dictionnaire erreurs.

Organisation des données dans Excel

- **Feuille Etudiants :**
  - Contient une ligne d'en-tête : NOM, PRÉNOM, EMAIL.
  - Les informations extraites sont ajoutées ligne par ligne.
- **Feuille Erreurs :**
  - Contient une ligne d'en-tête : Erreur, Nombre de fois.
  - Les erreurs et leur fréquence (extraites de erreurs) sont ajoutées ligne par ligne.

Sauvegarde et affichage

- Le fichier Excel est sauvegardé sous le nom Infos_Etudiants1.xlsx.
- Un message de confirmation est affiché indiquant la fin du traitement.

Lot 2 :

- **Structuration**
  - **Objectif :** Ce programme extrait des données structurées à partir d'un fichier Excel source (Infos_Etudiants.xlsx), les filtre et les organise dans un nouveau fichier Excel (Erreurs_Separees.xlsx). Il vise à traiter uniquement les lignes contenant des informations séparées par le délimiteur -.

- **Détails :**

Importation des modules

- **load_workbook et Workbook (openpyxl) :** Permettent de lire le fichier Excel source et de créer un fichier Excel de sortie.

Initialisation des fichiers

- **Fichier source :** Infos_Etudiants.xlsx, supposé être dans le même répertoire que le script.
- **Fichier de sortie :** Erreurs_Separees.xlsx, généré dans le répertoire courant.

Création du fichier de sortie

- Un nouveau classeur Excel est créé avec une feuille nommée Erreurs.
- Les en-têtes pour la feuille de sortie sont définis : DATE, NUMERO, TYPE, SPECIFICITE, REPONSE.

Définition de la fonction extraire_champs

- **Objectif :** Diviser une ligne en plusieurs champs en se basant sur le délimiteur -.
- Utilise la méthode **split** pour découper la ligne en parties à chaque occurrence de -.
- Supprime les espaces inutiles avec **strip** pour assurer un formatage propre.

Parcours des données du fichier source

- Les lignes du fichier source sont parcourues à partir de la deuxième ligne (en-têtes ignorés).
- Chaque cellule de la ligne est analysée individuellement :
    - Si elle contient une chaîne avec le délimiteur -, la fonction extraire_champs est utilisée pour diviser la donnée en champs.
    - Si au moins 5 champs sont extraits, ils sont ajoutés dans la feuille de sortie.
    - Sinon, la ligne est ignorée, et un message est affiché pour indiquer la raison.

Comptage des lignes traitées et ignorées

- **lignes_traitees :** Compte les lignes correctement formatées et ajoutées au fichier de sortie.
- **lignes_ignorees :** Compte les lignes non conformes (moins de 5 champs ou absence du délimiteur -).

Sauvegarde du fichier de sortie

- Les données traitées sont enregistrées dans un fichier Excel nommé Erreurs_Separees.xlsx.

Ajout d'un identifiant

- Une colonne intitulée id_type_erreur a été ajoutée dans le fichier Excel Erreurs_Separees.xlsx. Cette colonne attribue un identifiant unique, allant de 1 à 4, à chaque type d'erreur (par exemple, 1 pour SyntaxError, 2 pour TypeError, etc.). Cela permet de classifier les erreurs de manière claire et structurée.

Chat_Bot

- **Objectif :** Ce programme permet de rechercher des réponses spécifiques dans un fichier Excel (Erreurs_Separees.xlsx) en fonction d'un mot-clé choisi par l'utilisateur parmi une liste prédéfinie (SyntaxError, TypeError, NameError, IndentationError).
- **Détails :**

Définition du dictionnaire mots_cles

- **Contenu :** Associe chaque mot-clé à un identifiant numérique sous forme de chaîne de caractères :
  - SyntaxError -> "1"
  - TypeError -> "2"
  - NameError -> "3"
  - IndentationError -> "4"

- **Utilité :** Simplifie la correspondance entre les mots-clés et les identifiants présents dans le fichier Excel.

Saisie utilisateur

- L'utilisateur est invité à entrer un mot-clé parmi ceux définis dans le dictionnaire.
- Si le mot-clé saisi n'est pas valide (pas présent dans mots_cles), le programme affiche un message d'erreur et s'arrête.

Chargement du fichier Excel

- Le fichier Excel Erreurs_Separees.xlsx est chargé à l'aide de la bibliothèque **openpyxl**.
- La première feuille (active par défaut) est utilisée pour la recherche.

Recherche des réponses dans Excel

- **Objectif :** Trouver toutes les réponses associées à l'identifiant du mot-clé saisi.
- Les lignes sont parcourues en ignorant la première ligne (supposée contenir les en-têtes).
- Deux colonnes sont utilisées :
    - **6e colonne :** Contient les identifiants des mots-clés.
    - **5e colonne :** Contient les réponses associées.
- Si l'identifiant de la ligne correspond à celui du mot-clé, la réponse est ajoutée à un ensemble (set) pour éviter les doublons.

Affichage des résultats

- **Cas où des réponses sont trouvées :**
    - Le programme affiche les 20 premières réponses trouvées, numérotées de manière lisible.
- **Cas où aucune réponse n'est trouvée :**
    - Un message indique qu'aucune réponse n'a été trouvée pour le mot-clé donné.

4. Résultats obtenus

Lot 1 :

- **MailingList**
  - **Réponse affichée :** Extraction terminée. Les données ont été écrites dans mailingList.csv.
  - **Exemple les réponses affichées sur le fichier csv :**

NOM,PRENOM,ETUDIANTS EMAIL
YILDIZ,OLIVIER,olivier.yildiz@laposte.net
TOUCHARD,JOSETTE,josette.touchard@gmail.com
FREUND,BAPTISTE,baptiste.freund764@wanadoo.fr

- **Infos_Etudiants**
  - **Réponse affichée :** Extraction terminée. Les données ont été écrites dans Infos_Etudiants1.xlsx.
  - **Exemple les réponses affichées sur le fichier Excel :**
    - **Feuille Etudiants :**

NOM	PRENOM	EMAIL
YILDIZ	OLIVIER	olivier.yildiz@laposte.net
TOUCHARD	JOSETTE	josette.touchard@gmail.com
FREUND	BAPTISTE	baptiste.freund764@wanadoo.fr
MADELAINE	AH	ah.madelaine@free.fr

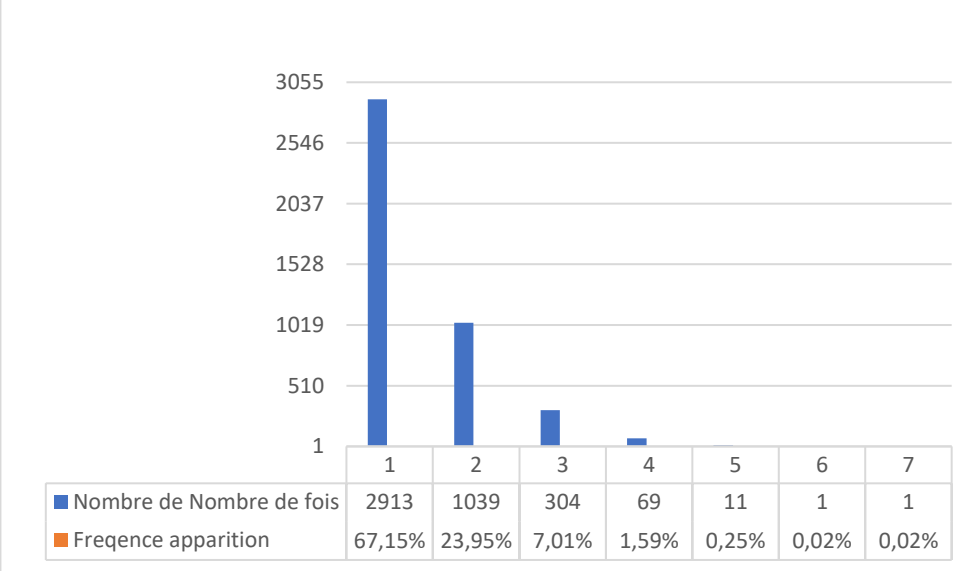
    - **Feuille Erreurs :**

Erreur	Nombre de fois
12/11/2024 - 1 - SyntaxError - unterminated string literal (detected at line 1) - fermer parenthèse	7
25/11/2024 - 3 - SyntaxError - invalid syntax for set literal - vérifier la syntaxe des ensembles	6
18/11/2024 - 3 - SyntaxError - invalid syntax wb.save ('nouveau_fichier.xlsx')s - enlever le 's'	5
4/09/2024 - 2 - NameError - name 'apend' is not defined. Did you mean 'append'? - corriger en append	5

- **Nombre de fois que les erreurs sont rencontrées dans les fichiers .txt avec leurs fréquences d'apparitions dans les fichiers (Tri à plat)**

Étiquettes de lignes	Nombre de Nombre de fois	Freqence apparition
1	2913	67,15%
2	1039	23,95%
3	304	7,01%
4	69	1,59%
5	11	0,25%
6	1	0,02%
7	1	0,02%
Total général	**4338**	**100,00%**

- **Représentation graphique**



Lot 2 :

- **Structuration**
  - **Réponse affichée :**

    Nombre de lignes traitées : 4338
    Nombre de lignes ignorées : 0
    Les données ont été extraites et sauvegardées dans le fichier Erreurs_Separees.xlsx.

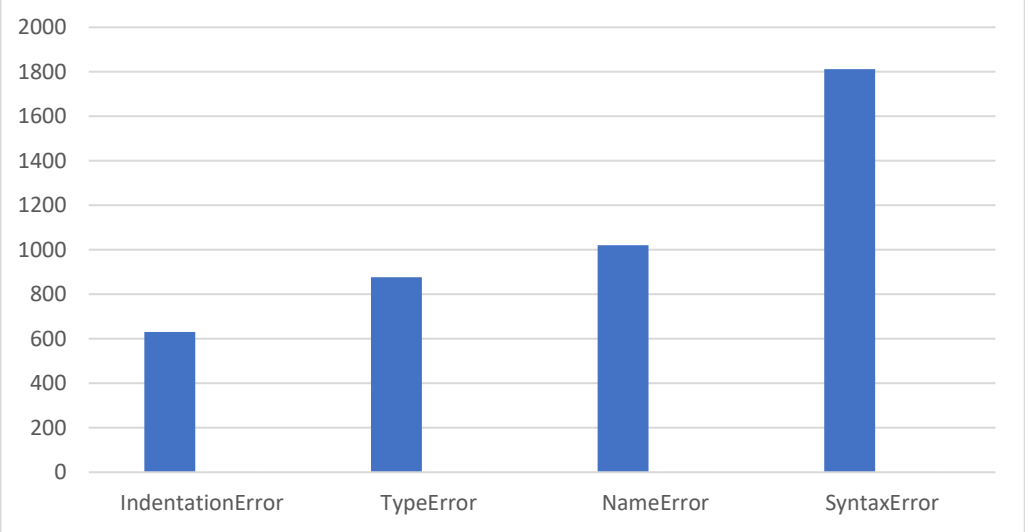
○ Exemple de réponses affichées sur le fichier Excel :

DATE	NUMERO	TYPE	SPECIFICITE	REPONSE	id_type_erreur
12/11/2024	1	SyntaxError	unterminated string literal (detected at line 1)	fermer parenthèse	1
25/11/2024	3	SyntaxError	invalid syntax for set literal	vérifier la syntaxe des ensembles	1
18/11/2024	3	SyntaxError	invalid syntax wb.save ('nouveau_fichier.xlsx')s	enlever le 's'	1
4/09/2024	2	NameError	name 'apend' is not defined. Did you mean 'append'?	corriger en append	3

○ Nombre d'erreurs par type d'erreur avec leurs fréquences d'apparitions (Tri à plat)

Étiquettes de lignes	Nombre de TYPE	Frequence apparution
IndentationError	630	14,52%
TypeError	876	20,19%
NameError	1020	23,51%
SyntaxError	1812	41,77%
Total général	**4338**	**100,00%**

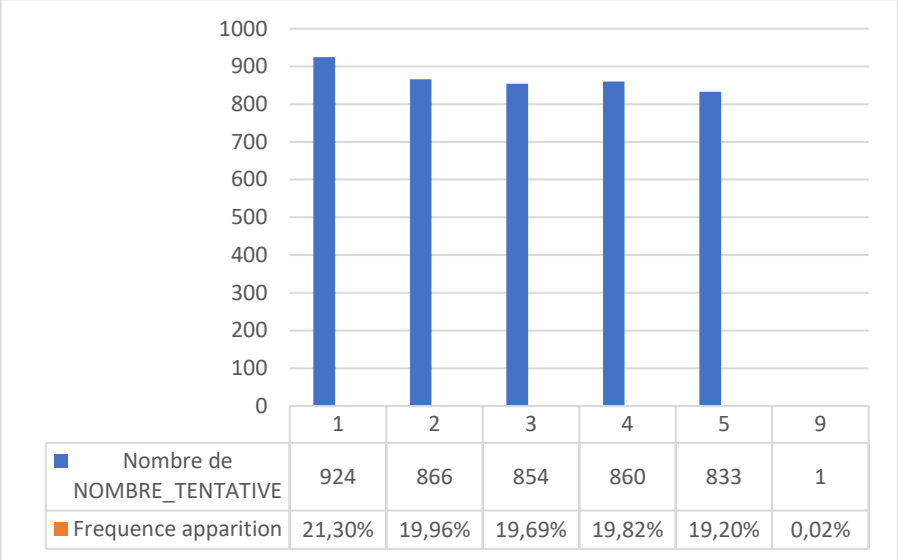
○ Représentation graphique



- **Nombre de tentatives avant réussite par étudiant (Tri à plat)**

Étiquettes de lignes	Nombre de NOMBRE_TENTATIVE	Frequence apparition
1	924	21,30%
2	866	19,96%
3	854	19,69%
4	860	19,82%
5	833	19,20%
9	1	0,02%
Total général	**4338**	**100,00%**

- **Représentation graphique**



Chat_Bot

- **Réponse affichée :**

Entrez un mot-clé parmi : SyntaxError, TypeError, NameError, IndentationError

Mot-clé : _

- **Exemple de réponses affichées dans les cas suivants :**
  - **Mot-clé :** SyntaxError

Voici quelques réponses pour le mot-clé 'SyntaxError':

1. vérifier la syntaxe des générateurs

2. vérifier les affectations

3. corriger l'indentation

4. écrire == au lieu de =

5. vrifier la syntaxe des ensembles

6. enlever le 's'

7. enlever la parenthèse en trop

8. enlever ou corriger la virgule

9. fermer parenthèse

10. changer '' par ""

11. vérifier la fermeture des blocs ou parenthèses

12. vérifier les caractères non valides dans les noms de variables

13. changer {} par ()

14. corriger la logique ou la syntaxe avec 'or'

15. éviter d'assigner des valeurs à des littéraux

16. placer 'continue' à l'intérieur d'une boucle

17. vérifier la syntaxe des ensembles

18. enlever la virgule

19. viter d'utiliser '=' dans une expression

20. quoted string literal

- **Mot-clé :** TypeError

Voici quelques réponses pour le mot-clé 'TypeError':

1. vérifier si une boucle tente d'itérer sur un entier

2. convertir la chaîne de caractères en entier avant d'utiliser

3. mettre les parenthèses

4. vérifier si on tente d'accéder à un type générique de manière incorrecte

5. 'str' and 'int'

6. utiliser une liste à la place d'un entier pour ajouter des éléments

7. dans print il y a un int et str

8. convertir un flottant en entier avant de l'utiliser comme index

9. ne mettre qu'un seul argument, on en a défini qu'un seul dans la fonction

10. mettre les parenthses

11. définir r

12. il faut donner une valeur a

13. vérifier si on tente d'accéder à un type générique de manière incorrecte

14. mettre en chaine de caractere pour pouvoir concatener

15. convertir un flottant en entier avant de l'utiliser comme index

16. mettre les parenthèses

17. vrifier si une boucle tente d'itrer sur un entier

18. vrifier si une fonction est correctement importe et appele

19. convertir la chaîne de caractères en entier avant d'utiliser

20. ne mettre qu'un seul argument, on en a dfini qu'un seul dans la fonction

- **Mot-clé :** NameError

Voici quelques réponses pour le mot-clé 'NameError':

1. écrire des valeurs pour x, y, z

2. changer la valeur

3. corriger en indent

4. corriger en exit

5. corriger en length

6. corriger en function

7. 'liste1'?

8. écrire True

9. écrire False

10. 's'?

11. corriger en __main__

12. corriger en config

13. corriger en __main__

14. corriger en exit

15. corriger en calculator

16. 'str'?

17. écrire des valeurs pour x, y, z

18. corriger en filter

19. corriger en config

20. corriger en receive

- **Mot-clé :** IndentationError

Voici quelques réponses pour le mot-clé 'IndentationError':

1. ajouter une indentation après finally

2. ajouter l'indentation aprs except

3. ajouter suite au lieu de u

4. ajouter l'indentation après except

5. decaller return

6. ajouter l'indentation après elif

7. ajouter l'indentation après else

8. ajouter l'indentation après try

9. ajouter l'indentation aprs else

10. ajouter espace avant

11. respecter les indentations

12. corriger l'indentation dans la compréhension de liste

13. enlever l'indentation

14. ajouter une indentation aprs return

15. ajouter l'indentation aprs while

16. vrifier les dductions dans le corps de la fonction

17. ajouter un bloc indenté après def

18. ajouter l'indentation après else

19. enlever l'indentation inattendue

20. erreur dans le nom de la clé

- **Mot-clé :** Error

Mot-clé invalide. Veuillez entrer un mot-clé valide.

5.Difficultés rencontrées et solutions apportées

Contrainte de temps :

Le délai pour terminer le projet était limité, ce qui a nécessité une gestion rigoureuse du temps pour respecter les échéances tout en assurant la qualité du travail.

Compréhension des sujets et des tâches :

Les consignes du projet n'étaient pas toujours claires dès le départ, notamment en ce qui concerne la structure attendue des fichiers et des données à extraire. Cela a demandé un effort supplémentaire pour clarifier les attentes.

Problèmes de formatage des données :

Lors de la génération des fichiers liés aux erreurs des étudiants, plusieurs cas particuliers ont été rencontrés dans les données :

- Certaines lignes comportaient des **':'** comme délimiteur au lieu de **-**. Pour résoudre ce problème, c'était modifié dans le code Python pour remplacer automatiquement les deux points par des tirets.
- Une ligne contenait un **espace simple** comme délimiteur (entre l'erreur et la réponse). Ce cas a été traité manuellement en remplaçant l'espace par un tiret dans les données avant de les réintégrer au fichier.

6.Conclusion

Ce projet a permis de développer des compétences pratiques en manipulation de données structurées avec Python et Excel, tout en mettant en évidence les défis liés au traitement de données réelles. Les contraintes de temps et les incohérences dans les fichiers source ont rendu le projet exigeant, mais elles ont également offert une expérience précieuse pour apprendre à résoudre des problèmes concrets.

Malgré les difficultés rencontrées, nous avons pu atteindre les objectifs fixés et produire des fichiers conformes aux attentes. Ce travail a renforcé notre capacité à nous adapter face à des données imparfaites et à améliorer nos méthodes pour une meilleure efficacité future.